G06_HW 2

Shashank Shekhar – (sshekha4)
Rahul Aettapu –(raettap)
Purva Vasudeo – (ppvasude)

Q1)

We need to calculate the best discrete split for the continuous attribute V1. We have done this using the entropy method. Please refer the excel sheet called q1_data_sets.csv on the page 'V1 split_Entropy'. Basically the values are arranged in non decreasing order. Then the entropy is calculated for each discrete split. The gain is also calculated. Eg. for the < 10 and >= 10 entropy is calculated as follows:

H(Class) = -(9/16)*lg(9/16) -(7/16)*lg(7/16)
= 0.4669 + 0.5217
= 0.9886

	10	
	<	>=
TRUE	0	7
FALSE	1	8
Entropy	0	0.99679
Gain	0.05411	

This can be read as if <10 values are considered for V1 in the dataset, then we have 0 True and 1 False.

H(v1<10) = 0 ….Since data is homogenous
H(v1 >= 10) = -(7/15)*lg(7/15) -(8/15)*lg(8/15) = 0.99679

Gain(Class/v1=10) = 0.9886 - (1/16)*0 - (15/16)*0.99679 = 0.05411

All other gains for the other discrete splits are also calculated in a similar way and appropriately put in the table in the excel sheet.

As v1=11 has the highest gain, we proceed with a discrete split for V1 as <11 and >=11.

a)
Please note, wherever I use the symbol 'lg(x)', it means log2(x). The character 'C' we are going to use as short for 'Class'.
Below is a snapshot of our data set :

V1	V2	V3	V4	V5	Class
7	BLUE	LONG	HOT	HIGH	FALSE
10	WHITE	SHORT	COOL	HIGH	FALSE
11	BLUE	SHORT	HOT	HIGH	TRUE
13	WHITE	LONG	HOT	HIGH	FALSE
15	BLUE	SHORT	COOL	HIGH	TRUE
18	WHITE	SHORT	HOT	HIGH	FALSE
20	BLUE	LONG	COOL	HIGH	FALSE
22	WHITE	LONG	COOL	HIGH	FALSE
27	WHITE	LONG	COOL	LOW	TRUE
30	BLUE	SHORT	COOL	LOW	TRUE
32	WHITE	SHORT	COOL	LOW	TRUE
35	BLUE	SHORT	HOT	LOW	TRUE
37	WHITE	SHORT	HOT	LOW	FALSE
40	BLUE	LONG	COOL	LOW	TRUE
43	WHITE	LONG	HOT	LOW	FALSE
50	BLUE	LONG	HOT	LOW	FALSE

Now we will calculate entropy of class :
H(C) = -9/16*lg(9/16) -7/16*lg(7/16)
= 0.4669 + 0.5217
= 0.989 …… (1)

H(C/V2) = 8/16*H(C/V2=Blue) + 8/16*H(C/V2=White)
H(C/V2=Blue) = -3/8*lg(3/18) -5/8*lg(5/8)
= 0.955

H(C/V2=White) = -6/8*lg(6/18) -2/8*lg(2/8)
= 0.811

G(C,V2) = 0.989 - H(C/V2)
= 0.989 - 0.5*0.955 - 0.5*0.811
= 0.106 …… (2)

H(C/V3) = 8/16*H(C/V3=Long) + 8/16*H(C/V3=Short)
H(C/V3=Short) = -3/8*lg(3/18) -5/8*lg(5/8)
= 0.955

H(C/V3=Long) = -6/8*lg(6/18) -2/8*lg(2/8)
= 0.811

G(C,V3) = 0.989 - H(C/V3)
= 0.989 - 0.5*0.955 - 0.5*0.811
= 0.106 …… (3)

H(C/V4) = 8/16*H(C/V4=Hot) + 8/16*H(C/V4=Cool)
H(C/V4=Cool) = -3/8*lg(3/18) -5/8*lg(5/8)
= 0.955

H(C/V4=Hot) = -6/8*lg(6/18) -2/8*lg(2/8)
= 0.811

G(C,V4) = 0.989 - H(C/V4)
= 0.989 - 0.5*0.955 - 0.5*0.811
= 0.106 …… (4)

H(C/V5) = 8/16*H(C/V5=High) + 8/16*H(C/V5=Low)
H(C/V5=Low) = -3/8*lg(3/18) -5/8*lg(5/8)
= 0.955

H(C/V5=High) = -6/8*lg(6/18) -2/8*lg(2/8)
= 0.811

G(C,V5) = 0.989 - H(C/V5)
= 0.989 - 0.5*0.955 - 0.5*0.811
= 0.106 …… (5)

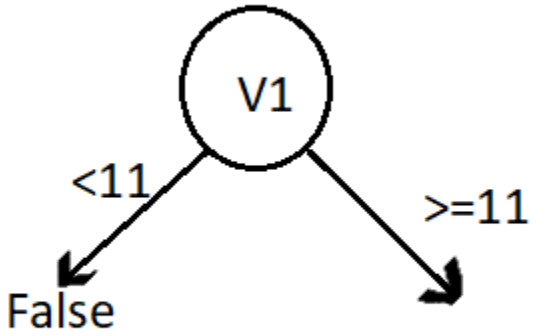
H(C/V1) = 2/16*H(C/V1= '< 11') + 14/16*H(C/V1= '>=11')
H(C/V1= '>=11') = -1/2*lg(1/2) -1/2*lg(1/2)
= 1

H(C/V1= '< 11') = 0 ..Data is homogenous

G(C,V1) = 0.989 - H(C/V1)
= 0.989 - (2/16)*0 - (14/16)*1
= 0.1136 …… (6)

Now we split on V1, as it gives us the highest gain :

Tree formed till now :



Now, the data set received which is heterogenous on >= 11 is

V2	V3	V4	V5	Class
BLUE	SHORT	HOT	HIGH	TRUE
WHITE	LONG	HOT	HIGH	FALSE
BLUE	SHORT	COOL	HIGH	TRUE
WHITE	SHORT	HOT	HIGH	FALSE
BLUE	LONG	COOL	HIGH	FALSE
WHITE	LONG	COOL	HIGH	FALSE
WHITE	LONG	COOL	LOW	TRUE
BLUE	SHORT	COOL	LOW	TRUE
WHITE	SHORT	COOL	LOW	TRUE
BLUE	SHORT	HOT	LOW	TRUE
WHITE	SHORT	HOT	LOW	FALSE
BLUE	LONG	COOL	LOW	TRUE
WHITE	LONG	HOT	LOW	FALSE
BLUE	LONG	HOT	LOW	FALSE

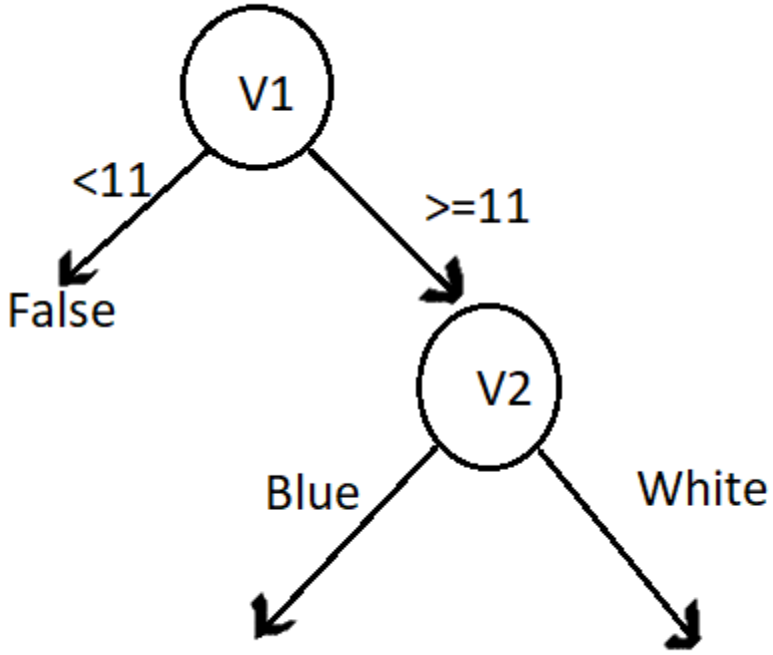
H(Class) = -7/14*lg(7/14) - 7/14*lg(7/14)
= 1 …….(7)

H(C/V3) = 7/14*H(C/V3=Long) + 7/14*H(C/V3=Short)

H(C/V3=Long) = -(5/7)lg(5/7) - (2/7)lg(2/7) = 0.8631
H(C/V3=Short) = -(5/7)lg(5/7) - (2/7)lg(2/7) = 0.8631
G(C,V3) = 1 - 1/2*(0.8631) - 1/2*(0.8631) = 0.1369

Similarly calculating,
G(C,V2) = 0.1369 …….(9)
G(C,V4) = 0.1369 …….(10)
G(C,V5) = 0.061 ……..(11)

Hence taking into consideration the highest gain, and since there is a tie, the leftmost attribute, we decide to split on V2.
Hence, tree formed till now is :



Now, reducing the data set even further, we have below data set to consider on the 'Blue' value of V2 :

V3	V4	V5	Class
SHORT	HOT	HIGH	TRUE
SHORT	COOL	HIGH	TRUE
LONG	COOL	HIGH	FALSE
SHORT	COOL	LOW	TRUE
SHORT	HOT	LOW	TRUE
LONG	COOL	LOW	TRUE
LONG	HOT	LOW	FALSE

H(C) = -(2/7)lg(2/7) - (5/7)lg(5/7) = 0.8631

H(C/V3) = 3/7*H(C/V3=Long) + 4/7*H(C/V3=Short)

H(C/V3=Long) = -(2/3)lg(2/3) - (1/3)lg(1/3) = 0.9183
H(C/V3=Short) = 0 …..As data is homogenous

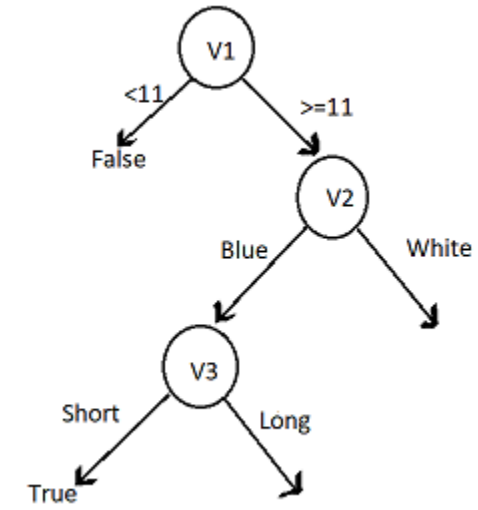
G(C,V3) = 0.8631 - 3/7*0.9183 + 4/7*0
= 0.4695

Similarly gains for other attributes are :

G(C,V4) = 0.00594
G(C,V5) = 0.00594

We now split on V3 as this gives us highest gain:

Hence now the decision tree looks like :



Now, the data set we get on the Long value of V3 is :

V4	V5	Class
COOL	HIGH	FALSE
COOL	LOW	TRUE
HOT	LOW	FALSE

H(C) = -2/3*lg(2/3)-1/3*lg(1/3)
= 0.9183 ……..(12)

H(C/V4) = 2/3*H(C/V4=Cool) + 1/3*H(C/V4=Hot)

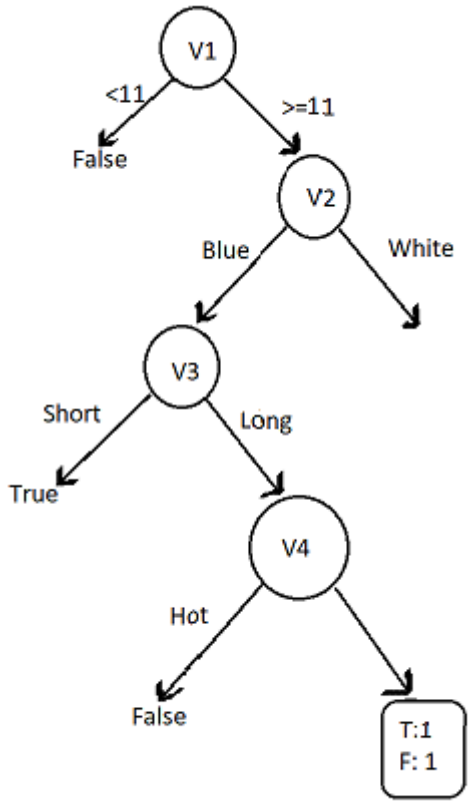
H(C/V4=Cool) = 1

H(C/V4=Hot) = 0

G(C,V4) = 0.9183 - 2/3*1 -1/3*(0)
= 0.516 …… (13)

Doing similar calculations,
G(C/V5) = 0.516

As there is a tie, we choose V4 to split on next and as we have reached the maximum depth , we stop:



Now the data set remaining on the White value of V2 is :

V3	V4	V5	Class
LONG	HOT	HIGH	FALSE
SHORT	HOT	HIGH	FALSE
LONG	COOL	HIGH	FALSE
LONG	COOL	LOW	TRUE
SHORT	COOL	LOW	TRUE
SHORT	HOT	LOW	FALSE
LONG	HOT	LOW	FALSE

H(C) = -(2/7)lg(2/7) - (5/7)lg(5/7) = 0.8631
H(C/V3) = 4/7*H(C/V3=Long) + 3/7*H(C/V3=Short)

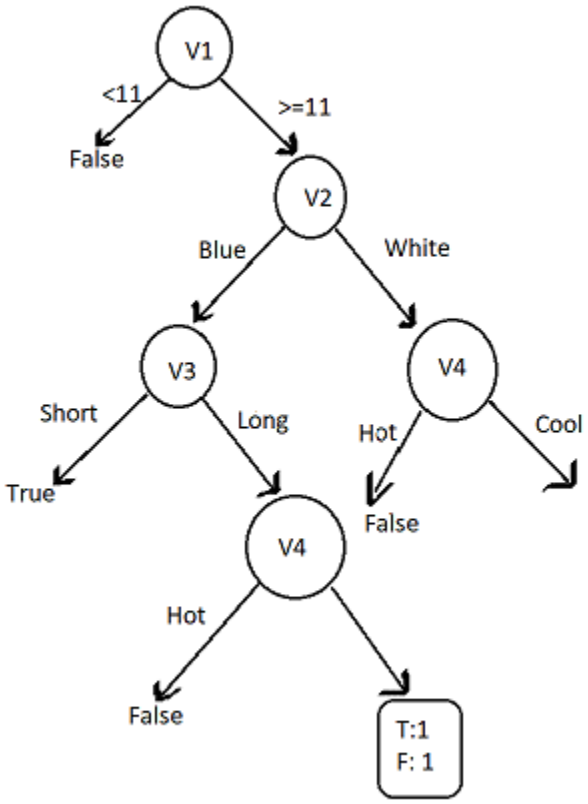
H(C/V3=Long) = -3/4*lg(3/4)-1/4*lg(1/4)
= 0.8113

H(C/V3=Short) = = -(2/3)lg(2/3) - (1/3)lg(1/3) = 0.9183

G(C,V3) = 0.8631 - (4/7)*0.8113 - (3/7)*0.9183 = 0.005943

Similarly,
G(C,V4) = 0.4695
G(C,V5) = 0.2917

As V4 has highest gain we split on that :



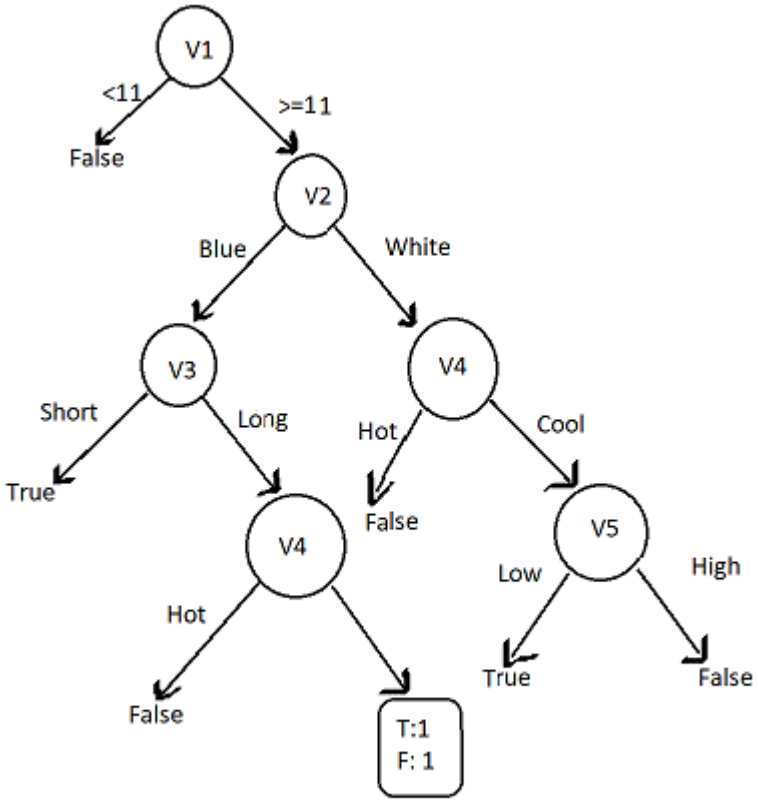
Now the data set on V4= cool is :

V3	V5	Class
LONG	HIGH	FALSE
LONG	LOW	TRUE
SHORT	LOW	TRUE

H(C) = -1/3*lg(1/3) -2/3*lg(2/3) = 0.9183
H(C/V5) = 1/3*H(C/V5=High) + 2/3*H(C/V5=Low)
H(C/V5=Low) = 0 ...Data is Homogenous
H(C/V5=High) = 0 ...Data is Homogenous

G(C,V5) = 0.9183 - 0 = 0.9183

As this is the highest gain we can get, we now split on V5.



b)

We need to calculate the best discrete split for the continuous attribute V1. We have done this using the gini index method. Please refer the excel sheet called q1_data_sets.csv on the page 'V1 split_GINI'. Basically the values are arranged in non decreasing order. Then the average of each successive pair of values is calculated. Eg. Average of 7 and 10 is 8. Then the number of data points corresponding to a 'True' and 'False' are mentioned for each <= and > than each of the average values. Then the gini index is calculated for each one. Eg. Gini index if we split this continuous attribute on a discrete value of 8 is 0.467. Calculation was done as below for average value 8:

Average	8	
	<=	>
TRUE	0	7
FALSE	1	8

GINI(<= 8) = $1 - (0/1)^2 - (1/1)^2 = 0$

$GINI(>8) = 1 - (7/15)^2 - (8/15)^2 = 0.4978$
$GINI_{split}(8) = (1/16)*0 + (15/16)*0.4978 = 0.467$

All other GINI calculations are also done in the same manner. Now, we look for the GINI split index which gives us the least value. Looking at the table, this value is 0.42 which arises if we split at 24. Hence, the V1 continuous attribute we will be splitting on 24, with <=24 and >24.

Now we will proceed with the further calculations to get the best attribute to split on for the decision tree.

The gini split for V1 is as below :

Class Values	V1	
	<= 24	> 24
T	2	5
F	6	3
	0.42	

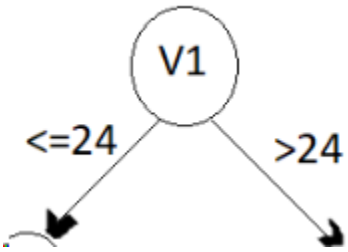
$Gini_s(V1) = 0.422$

Class Values	V2	
	B	W
T	5	2
F	3	6
	0.42	

$G(B) = 1 - (5/8)^2 - (3/8)^2 = 0.469$
$G(W) = 1 - (2/8)^2 - (6/8)^2 = 0.375$
$Gini_s(V2) = 0.469*8/16 + 0.375*8/16$
$Gini_s(V2) = 0.422$

Similarly other gini splits are,
$Gini_s(V3) = 0.422$
$Gini_s(V4) = 0.422$
$Gini_s(V5) = 0.422$

Now, we choose the lowest gini split valued attribute to split on in the decision tree. As there is a tie, we choose the leftmost attribute. Now the decision tree looks like :



Now we will look at the <= 24 side, data set to consider is :

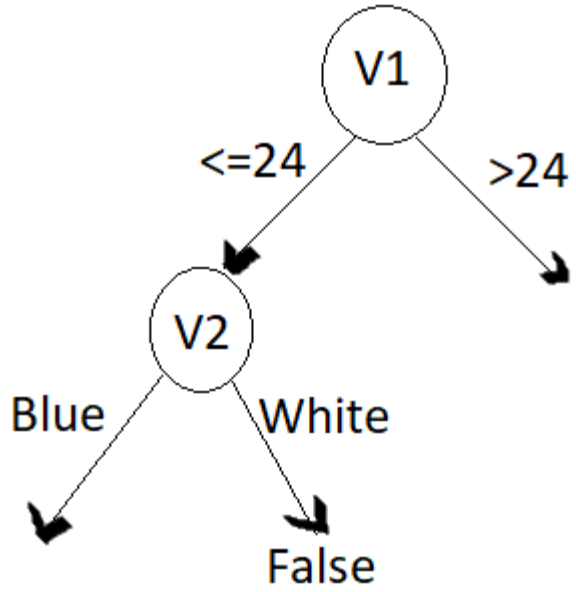
V2	V3	V4	V5	Class
BLUE	LONG	HOT	HIGH	FALSE
WHITE	SHORT	COOL	HIGH	FALSE
BLUE	SHORT	HOT	HIGH	TRUE
WHITE	LONG	HOT	HIGH	FALSE
BLUE	SHORT	COOL	HIGH	TRUE
WHITE	SHORT	HOT	HIGH	FALSE
BLUE	LONG	COOL	HIGH	FALSE
WHITE	LONG	COOL	HIGH	FALSE

Class Values	V2	
	B	W
T	2	0
F	2	4
	0.25	

$G(B) = 1 - (2/4)^2 - (2/4)^2 = 0.5$
$G(W) = 1 - (0/4)^2 - (4/4)^2 = 0$
$Gini_s(V2) = 0.5*4/8 + 0*4/8$
$Gini_s(V2) = 0.25$

Similarly, other attribute gini splits are as below :
$Gini_s(V3) = 0.25$
$Gini_s(V4) = 0.375$
$Gini_s(V5) = 0.375$

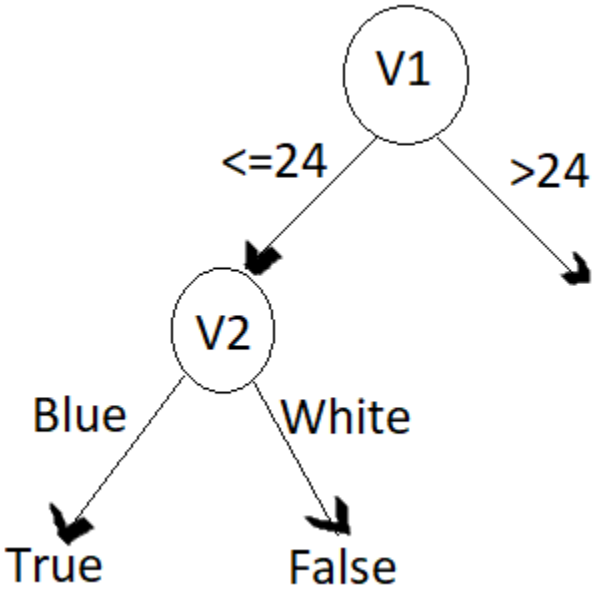
Now we according to above values , we choose V2 to split on next :



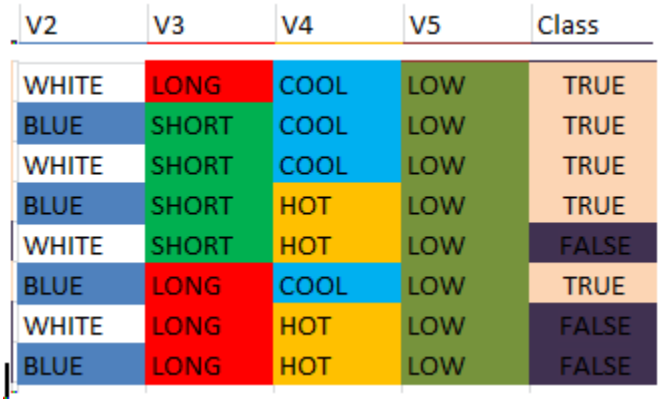
As we have reached the maximum depth possible for the tree i.e. 2, we need to decide a value from class for the 'Blue' decision. We see the data we have left which might reach this node :

V3	V4	V5	Class
LONG	HOT	HIGH	FALSE
SHORT	HOT	HIGH	TRUE
SHORT	COOL	HIGH	TRUE
LONG	COOL	HIGH	FALSE

As there is equal number of True and False possible, we randomly choose the value to be given here as 'True'. Hence now the tree looks like :



Now going to the >24 data set we have :



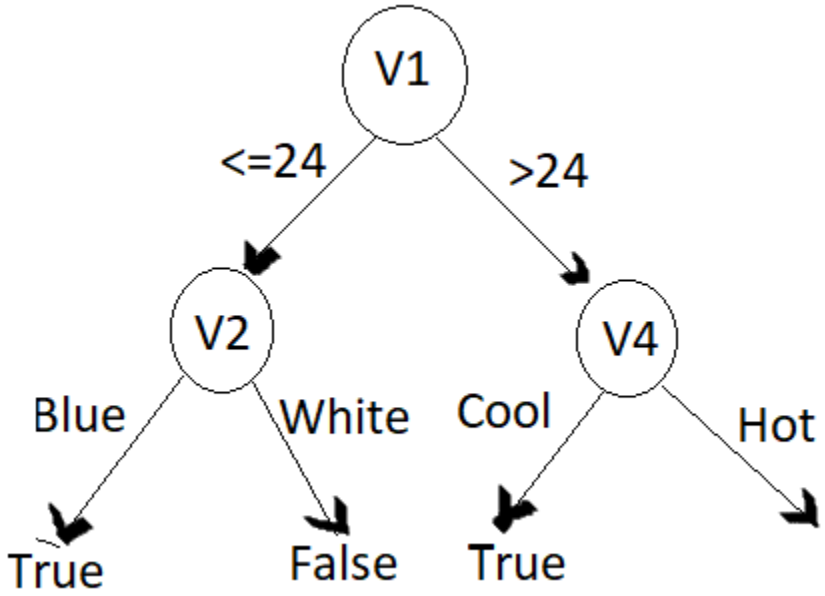
V2	V3	V4	V5	Class
WHITE	LONG	COOL	LOW	TRUE
BLUE	SHORT	COOL	LOW	TRUE
WHITE	SHORT	COOL	LOW	TRUE
BLUE	SHORT	HOT	LOW	TRUE
WHITE	SHORT	HOT	LOW	FALSE
BLUE	LONG	COOL	LOW	TRUE
WHITE	LONG	HOT	LOW	FALSE
BLUE	LONG	HOT	LOW	FALSE

Class Values	V2	
	B	W
T	3	2
F	1	2
	0.25	

$G(B) = 1 - (3/4)^2 - (1/4)^2 = 0.375$
$G(W) = 1 - (2/4)^2 - (2/4)^2 = 0,5$
$Gini_s(V2) = 0.5*4/8 + 0.375*4/8$
$Gini_s(V2) = 0.438$

Similarly other gini splits for the remaining attributes are:
$Gini_s(V3) = 0.438$
$Gini_s(V4) = 0.375$
$Gini_s(V5) = 0.469$

As V4 has lowest split value, we choose V4 as an attribute to split on next :

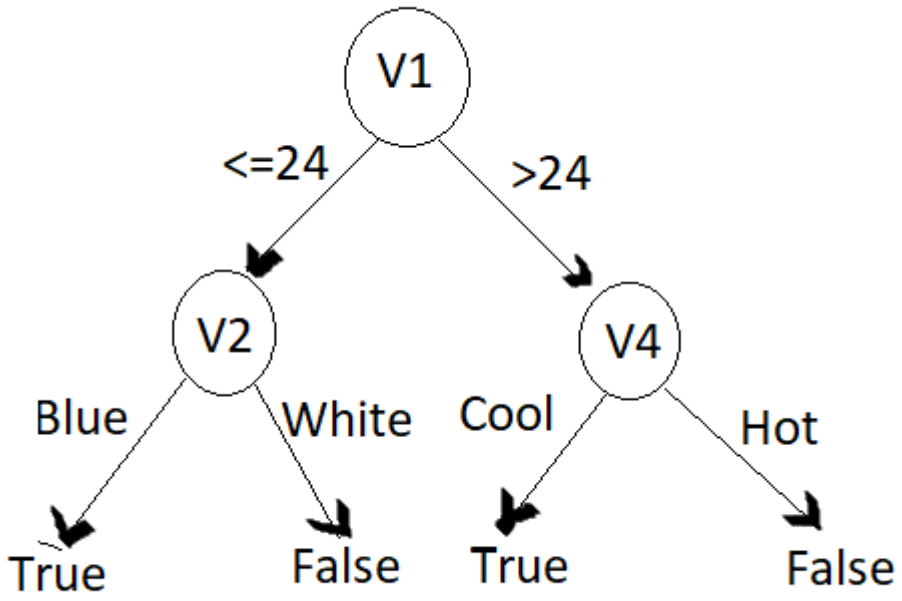


The data set we now have left is :

V2	V3	V5	Class
BLUE	SHORT	LOW	TRUE
WHITE	SHORT	LOW	FALSE
WHITE	LONG	LOW	FALSE
BLUE	LONG	LOW	FALSE

But we have reached the maximum depth. As we can see now in this data set, we have 3 False and 1 True, hence we classify all the data having value as Hot on the V4 node as False.

Final decision tree is :



c)

The difference in these two trees arises mainly due to the maximum depth permissible for each of them. The tree formed using ID3 has maximum depth as 4 which makes it more suited at least to the training data classification, while the tree formed using GINI has maximum depth as 2. If we just use the training data on both the trees, we can easily see that the ID3 tree with depth 4 will classify almost all of that training data correctly. On the other hand, the GINI tree might not be able to achieve this level of training data classification accuracy. Eg. take the below data into consideration from the training set :

V1	V2	V3	V4	V5	Class
35	BLUE	SHORT	HOT	LOW	TRUE

For the ID3 decision tree, the process for classifying this will be on the V1 node, as 35 > 11, it will go right, then on the V2 node as it is Blue, it will go left, on the V3 node as it is Short it will go left to give class value as True, which is the correct class value.

For the Gini decision tree, the process for classifying this will be on the V1 node, as 35 > 24, it will go right,then on the V4 node as it is Hot it will go right and classify this data as False, which is not its actual class as given. This happens because as the Gini tree stops at level 2, we had to give the highest probability class to the unexplored nodes. In this case, On V4=Hot we had 4 data points with 3 False and 1 True. As False had higher probability we labeled this leaf as False.

d)

As shown in the previous answer, the ID3 decision tree will perform better on the training data set, as it is perfectly fitted for this data and is made by considering the whole data set with just one leaf having equal chance of getting a True or False. The Gini tree might not perform as well on the training data as the ID3 tree, as we have a maximum depth hyperparameter on that one,

hence we had to make some probability based decisions on the leaf nodes and their class values. For the test data set, we cannot actually predict which will fit the data set better, It might be the case that the ID3 decision tree is overfitted to the training data and will hence fail on the test data. But as we do not actually know what the test data is, we cannot decide which tree will be better. Also, there is one leaf on the ID3 with equal probabilities of True/False, hence we cannot really label that one with one particular class label without having further information.

Q2)
a)
In the given decision tree, total number of misclassified instances are 8.
Optimistic Error = 8/34 = 0.235

For pessimistic error, the number of leaf nodes are 7 and there is a 0.5 penalty per leaf node.
Pessimistic Error = (8+0.5*7)/34 = 0.338

b)
The confusion matrix formed by classifying the test data using the decision tree is :

	Predicted: Yes	Predicted: No
n=20		
Actual: Yes	12	2
Actual: No	4	2

Please note,
TP : True Positive = 12
TN: True Negative = 2
FP: False Positive = 4
FN: False Negative = 2

Now,

Accuracy = (TP+TN)/(TP+FP+FN+TN)
= 14/20
= 0.7

Precision = TP/(TP+FP)
= 12/16
= 0.75

Recall = TP/(TP+FN)
= 12/14
= 0.857

F1 Score = 2*(Recall * Precision) / (Recall + Precision)
= 2*(0.857*0.75)/(0.857 + 0.75)
= 0.7999

Error Rate = (FP+FN)/(FP+FN+TP+TN) = 1 - Accuracy
= 1 - 0.7
= 0.3

Q3)

In the given decision tree, total number of misclassified instances before splitting Color are 10.
Optimistic Error = 10/34 = 0.294
In the given decision tree, total number of misclassified instances after splitting Color are 8.
Optimistic Error = 8/34 = 0.235
If we were to minimize the optimistic error rate then we should not prune the Color node as the optimistic error rate after splitting is less than the optimistic error rate before splitting.

b) The error factor of the node is 0.8
In the given decision tree, total number of misclassified instances before splitting Color are 10. Number of leaf nodes is 4.
Pessimistic Error = (10 + 4*0.8)/34 = 0.388

In the given decision tree, total number of misclassified instances after splitting Color are 8. Number of leaf nodes is 7.
Pessimistic Error = (8+7*0.8)/34 = 0.4

If we were to minimize the pessimistic error rate then we should prune the Color node as the pessimistic error rate after splitting is greater than the pessimistic error rate before splitting.

c) The confusion matrix formed by classifying the test data with the Color node pruned using the decision tree is :

	Predicted: Yes	Predicted: No
n=20		
Actual: Yes	13	1
Actual: No	6	0

Please note,
TP : True Positive = 13
TN: True Negative = 1
FP: False Positive = 6
FN: False Negative = 0

Error Rate = (FP+FN)/(FP+FN+TP+TN)
= 7/20
= 0.35

The Test Error Rate before the Color node pruned is calculated in Q2 which was found to be 0.30.
The training error before the Color node is pruned was 0.235. As the errors are almost likely and close to each other we cannot say that the original model was overfitted.

Q4)

ID	X1	X2	Y
1	35.0	15.0	-
2	2.5	11.0	-
3	10.5	12.5	+
4	44.0	11.0	+
5	1.5	13.0	-
6	48.0	11.0	+
7	45.0	13.0	-
8	38.0	10.0	+
9	7.5	13.5	-

a) Distance Matrix:

	1	2	3	4	5	6	7	8	9
1	0	32.7452	24.6272	9.84886	33.5596	13.6015	10.198	5.83095	27.5409
2	32.7452	0	8.13941	41.5	2.23607	45.5	42.547	35.5141	5.59017
3	24.6272	8.13941	0	33.5336	9.01388	37.53	34.5036	27.6137	3.16228
4	9.84886	41.5	33.5336	0	42.547	4.0	2.23607	6.0827	36.5855
5	33.5596	2.23607	9.01388	42.547	0	46.543	43.5	36.6231	6.0208
6	13.6015	45.5	37.53	4.0	46.543	0	3.60555	10.0499	40.5771
7	10.198	42.547	34.5036	2.23607	43.5	3.60555	0	7.61577	37.5033
8	5.83095	35.5141	27.6137	6.0827	36.6231	10.0499	7.61577	0	30.7002
9	27.5409	5.59017	3.16228	36.5855	6.0208	40.5771	37.5033	30.7002	0

b) Evaluating 1-NN Classifier

i) Holdout Method

ID	1	2	3	4	5	6	7	8	9
1	0	32.7452	24.6272	9.84886	33.5596	13.6015	10.198	5.83095	27.5409
2	32.7452	0	8.13941	41.5	2.23607	45.5	42.547	35.5141	5.59017
3	24.6272	8.13941	0	33.5336	9.01388	37.53	34.5036	27.6137	3.16228
4	9.84886	41.5	33.5336	0	42.547	4.0	2.23607	6.0827	36.5855
5	33.5596	2.23607	9.01388	42.547	0	46.543	43.5	36.6231	6.0208
6	13.6015	45.5	37.53	4.0	46.543	0	3.60555	10.0499	40.5771
7	10.198	42.547	34.5036	2.23607	43.5	3.60555	0	7.61577	37.5033
8	5.83095	35.5141	27.6137	6.0827	36.6231	10.0499	7.61577	0	30.7002
9	27.5409	5.59017	3.16228	36.5855	6.0208	40.5771	37.5033	30.7002	0

Testing Accuracy:

ID	1-NN	True Value	Predicted Value
6	4	+	+
7	4	-	+
8	1	+	-
9	3	-	+

Confusion Matrix:

	Predicted (Yes)	Predicted (No)
Actual (Yes)	1	1
Actual (No)	2	0

Accuracy = (True Positive + False Negative) / (True Positive + True Negative + False Positive + False Negative)

Accuracy = ¼ = 0.25

ii) 3 - fold Cross Validation

- 1st fold: [3,6,9]

	1	2	3	4	5	6	7	8	9
1	0	32.7452	24.6272	9.84886	33.5596	13.6015	10.198	5.83095	27.5409
2	32.7452	0	8.13941	41.5	2.23607	45.5	42.547	35.5141	5.59017
3	24.6272	8.13941	0	33.5336	9.01388	37.53	34.5036	27.6137	3.16228
4	9.84886	41.5	33.5336	0	42.547	4.0	2.23607	6.0827	36.5855
5	33.5596	2.23607	9.01388	42.547	0	46.543	43.5	36.6231	6.0208
6	13.6015	45.5	37.53	4.0	46.543	0	3.60555	10.0499	40.5771
7	10.198	42.547	34.5036	2.23607	43.5	3.60555	0	7.61577	37.5033
8	5.83095	35.5141	27.6137	6.0827	36.6231	10.0499	7.61577	0	30.7002
9	27.5409	5.59017	3.16228	36.5855	6.0208	40.5771	37.5033	30.7002	0

Testing Accuracy:

ID	1-NN	True Value	Predicted Value
3	2	+	-
6	7	+	-
9	2	-	-

- 2nd fold: [1,4,7]

	1	2	3	4	5	6	7	8	9
1	0	32.7452	24.6272	9.84886	33.5596	13.6015	10.198	5.83095	27.5409
2	32.7452	0	8.13941	41.5	2.23607	45.5	42.547	35.5141	5.59017
3	24.6272	8.13941	0	33.5336	9.01388	37.53	34.5036	27.6137	3.16228
4	9.84886	41.5	33.5336	0	42.547	4.0	2.23607	6.0827	36.5855
5	33.5596	2.23607	9.01388	42.547	0	46.543	43.5	36.6231	6.0208
6	13.6015	45.5	37.53	4.0	46.543	0	3.60555	10.0499	40.5771
7	10.198	42.547	34.5036	2.23607	43.5	3.60555	0	7.61577	37.5033

8	5.83095	35.5141	27.6137	6.0827	36.6231	10.0499	7.61577	0	30.7002
9	27.5409	5.59017	3.16228	36.5855	6.0208	40.5771	37.5033	30.7002	0

Testing Accuracy:

ID	1-NN	True Value	Predicted Value
1	8	-	+
4	6	+	+
7	6	-	+

- 3rd fold: [2,5,8]

	1	2	3	4	5	6	7	8	9
1	0	32.7452	24.6272	9.84886	33.5596	13.6015	10.198	5.83095	27.5409
2	32.7452	0	8.13941	41.5	2.23607	45.5	42.547	35.5141	5.59017
3	24.6272	8.13941	0	33.5336	9.01388	37.53	34.5036	27.6137	3.16228
4	9.84886	41.5	33.5336	0	42.547	4.0	2.23607	6.0827	36.5855
5	33.5596	2.23607	9.01388	42.547	0	46.543	43.5	36.6231	6.0208
6	13.6015	45.5	37.53	4.0	46.543	0	3.60555	10.0499	40.5771
7	10.198	42.547	34.5036	2.23607	43.5	3.60555	0	7.61577	37.5033
8	5.83095	35.5141	27.6137	6.0827	36.6231	10.0499	7.61577	0	30.7002
9	27.5409	5.59017	3.16228	36.5855	6.0208	40.5771	37.5033	30.7002	0

Test Accuracy:

ID	1-NN	True Value	Predicted Value
2	9	-	-
5	9	-	-
8	1	+	-

Confusion Matrix:

	Predicted (Yes)	Predicted (No)
Actual (Yes)	1	3
Actual (No)	2	3

Accuracy = (True Positive + False Negative) / (True Positive + True Negative + False Positive + False Negative)

Accuracy = 4/9 = 0.56

iii) LOOCV

	1	2	3	4	5	6	7	8	9
1	0	32.7452	24.6272	9.84886	33.5596	13.6015	10.198	5.83095	27.5409
2	32.7452	0	8.13941	41.5	2.23607	45.5	42.547	35.5141	5.59017
3	24.6272	8.13941	0	33.5336	9.01388	37.53	34.5036	27.6137	3.16228
4	9.84886	41.5	33.5336	0	42.547	4.0	2.23607	6.0827	36.5855
5	33.5596	2.23607	9.01388	42.547	0	46.543	43.5	36.6231	6.0208
6	13.6015	45.5	37.53	4.0	46.543	0	3.60555	10.0499	40.5771
7	10.198	42.547	34.5036	2.23607	43.5	3.60555	0	7.61577	37.5033
8	5.83095	35.5141	27.6137	6.0827	36.6231	10.0499	7.61577	0	30.7002
9	27.5409	5.59017	3.16228	36.5855	6.0208	40.5771	37.5033	30.7002	0

Computing the closest distance of each node from the other nodes and hence building the test accuracy table:

Test Accuracy:

ID	1-NN	True Value	Predicted Value
1	8	-	+
2	5	-	-
3	2	+	-
4	7	+	-
5	2	-	-
6	7	+	-
7	6	-	+
8	1	+	-
9	3	-	+

Confusion Matrix:

	Predicted (Yes)	Predicted (No)
Actual (Yes)	0	4
Actual (No)	3	2

Accuracy = (True Positive + False Negative) / (True Positive + True Negative + False Positive + False Negative)

Accuracy = 2/9

c) Simple Majority Classifier performs differently because there are equal instances of both the classes (+ and -). As a result of this on using the simple majority classifier always classifies wrongly. Hence the accuracy will always be zero. For example: there are 10 positive classes and 9 negative classes and one of the negative class is taken as the validation set. Now since initially there was an equal distribution, now there will be an imbalance and the label would be wrongly classified always into the opposite class label. This would happen for all the iterations and hence the accuracy will be zero.

Q5)

a) The confusion matrix and accuracy of the different types of classifiers are mentioned below:

Euclidean KNN:

```
              Reference
Prediction 1 2 3 4
         1 8 5 7 6
         2 0 9 1 0
         3 0 0 6 0
         4 1 1 1 5
 Accuracy ->  0.56
```

Cosine KNN:

```
        Reference
Prediction  1  2  3  4
         1  9  0  1  2
         2  0 15  1  1
         3  0  0 13  1
         4  0  0  0  7
Accuracy -> 0.88
```

Confidence KNN:

```
        y_true
y_pred  1  2  3  4
```

```
     1  9  0  0  1
     2  0 14  1  1
     3  0  0 14  1
     4  0  1  0  8
Accuracy -> 0.9
```

Decision Tree:

```
          Reference
Prediction 1 2 3 4
         1 8 6 7 5
         2 0 8 0 0
         3 0 1 8 1
         4 1 0 0 5
Accuracy -> 0.58
```

Decision Tree after Tuning:

```
          Reference
Prediction 1 2 3 4
         1 8 6 7 5
         2 0 8 0 0
         3 0 1 8 1
         4 1 0 0 5
Accuracy -> 0.58
```

In terms of accuracy, Confidence KNN classifier has the highest accuracy. Euclidean KNN classifi er has the accuracy.

b) The misclassification errors for each class in each classifier are as belows:

Euclidean KNN:

Class	Miss Classifications
1	1
2	6
3	9
4	6

Class 3 has maximum misclassifications.

Cosine KNN:

Class	Miss Classifications
1	0
2	0
3	2
4	4

Class 4 has maximum misclassifications.

Confidence KNN:

Class	Miss Classifications
1	0
2	1
3	1
4	3

Class 4 has maximum misclassifications.

Decision Tree:

Class	Miss Classifications
1	1
2	7
3	7
4	6

Class 2 and 3 has maximum misclassifications.

Decision Tree With Tuning:

Class	Miss Classifications
1	1
2	7
3	7
4	6

Class 2 and 7 has maximum misclassifications.

➔ To check which class performs better than other class in each classifier we need to check out the precision and recall of each class. F1 measure gives us the harmonic mean of precision and recall helping us find out which class outperforms the other.

Euclidean KNN:

Statistics by Class:

	Class: 1	Class: 2	Class: 3	Class: 4
Precision	0.3077	0.9000	1.0000	0.6250
Recall	0.8889	0.6000	0.4000	0.4545
F1	0.4571	0.7200	0.5714	0.5263

Class 2 performs better.

Cosine KNN:

	Class: 1	Class: 2	Class: 3	Class: 4
Precision	0.7500	0.8824	0.9286	1.0000
Recall	1.0000	1.0000	0.8667	0.6364
F1	0.8571	0.9375	0.8966	0.7778

Class 2 performs better.

Confidence KNN:

	Class: 1	Class: 2	Class: 3	Class: 4
Precision	0.9000	0.8750	0.9333	0.8889
Recall	1.0000	0.9333	0.9333	0.7273
F1	0.9474	0.9032	0.9333	0.8000

Class 1 performs better.

Decision Tree:

	Class: 1	Class: 2	Class: 3	Class: 4
Precision	0.3077	1.0000	0.8000	0.8333
Recall	0.8889	0.5333	0.5333	0.4545
F1	0.4571	0.6957	0.6400	0.5882

Class 2 performs better.

Decision Tree with Tuning:

	Class: 1	Class: 2	Class: 3	Class: 4
Precision	0.3077	1.0000	0.8000	0.8333
Recall	0.8889	0.5333	0.5333	0.4545
F1	0.4571	0.6957	0.6400	0.5882

Class 2 performs better.

Overall Class 1 has less number of misclassifications and Class 2 performs better.